Python tutorial and Backpropagation

SETH HUANG

PYTHON TUTORIALS PART 3
WE ARE ALREADY MORE THAN HALF WAY THERE

"DEF:" PRACTICE: MAKE YOUR OWN FUNCTIONS

```
1  print "Let's practice everything."
2  print 'You\'d need to know \'bout escapes with \\ that do \n newlines and \t tabs.'
3
4  poem = """
5  \tThe lovely world
6  with logic so firmly planted
7  cannot discern \n the needs of love
8  nor comprehend passion from intuition
9  and requires an explanation
10 \n\t\twhere there is none.
11 """
12
13 print "--------------"
14 print poem
15 print "--------------"
16
17
18 five = 10 - 2 + 3 - 6
19 print "This should be five: %s" % five
20
21 def secret_formula(started):
22     jelly_beans = started * 500
23     jars = jelly_beans / 1000
24     crates = jars / 100
25     return jelly_beans, jars, crates
26
27
28 start_point = 10000
29 beans, jars, crates = secret_formula(start_point)
30
31 print "With a starting point of: %d" % start_point
32 print "We'd have %d beans, %d jars, and %d crates." % (beans, jars, crates)
33
34 start_point = start_point / 10
35
36 print "We can also do that this way:"
37 print "We'd have %d beans, %d jars, and %d crates." % secret_formula(start_point)
```

BOOLEAN LOGIC
DAD, “IS IT A BOY OR A GIRL?”
BOOLEAN DOCTOR, “YES!”

```
1. True and True
2. False and True
3. 1 == 1 and 2 == 1
4. "test" == "test"
5. 1 == 1 or 2 != 1
6. True and 1 == 1
7. False and 0 != 0
8. True or 1 == 1
9. "test" == "testing"
10. 1 != 0 and 2 == 1
11. "test" != "testing"
12. "test" == 1
13. not (True and False)
14. not (1 == 1 and 0 != 1)
15. not (10 == 1 or 1000 == 1000)
16. not (1 != 10 or 3 == 4)
17. not ("testing" == "testing" and "Zed" == "Cool Guy")
18. 1 == 1 and (not ("testing" == 1 or 1 == 0))
19. "chunky" == "bacon" and (not (3 == 4 or 3 == 3))
20. 3 == 3 and (not ("testing" == "testing" or "Python" == "Fun"))
```

WHAT IF? CONDITIONAL PROGRAMMING

```
people = 20
cats = 30
dogs = 15


if people < cats:
    print "Too many cats! The world is doomed!"

if people > cats:
    print "Not many cats! The world is saved!"

if people < dogs:
    print "The world is drooled on!"

if people > dogs:
    print "The world is dry!"


dogs += 5

if people >= dogs:
    print "People are greater than or equal to dogs."

if people <= dogs:
    print "People are less than or equal to dogs."


if people == dogs:
    print "People are dogs."
```

ELSEIF: ELSE AND IF

```
1   people = 30
2   cars = 40
3   trucks = 15
4
5
6   if cars > people:
7       print "We should take the cars."
8   elif cars < people:
9       print "We should not take the cars."
10  else:
11      print "We can't decide."
12
13  if trucks > cars:
14      print "That's too many trucks."
15  elif trucks < cars:
16      print "Maybe we could take the trucks."
17  else:
18      print "We still can't decide."
19
20  if people > trucks:
21      print "Alright, let's just take the trucks."
22  else:
23      print "Fine, let's stay home then."
```

YOU CAN NOW MAKE A (SOMEWHAT VIOLENT) GAME

```
1  print "You enter a dark room with two doors.  Do you go through door #1 or door #2?"
2
3  door = raw_input("> ")
4
5  if door == "1":
6      print "There's a giant bear here eating a cheese cake.  What do you do?"
7      print "1. Take the cake."
8      print "2. Scream at the bear."
9
10     bear = raw_input("> ")
11
12     if bear == "1":
13         print "The bear eats your face off.  Good job!"
14     elif bear == "2":
15         print "The bear eats your legs off.  Good job!"
16     else:
17         print "Well, doing %s is probably better.  Bear runs away." % bear
18
19 elif door == "2":
20     print "You stare into the endless abyss at Cthulhu's retina."
21     print "1. Blueberries."
22     print "2. Yellow jacket clothespins."
23     print "3. Understanding revolvers yelling melodies."
24
25     insanity = raw_input("> ")
26
27     if insanity == "1" or insanity == "2":
28         print "Your body survives powered by a mind of jello.  Good job!"
29     else:
30         print "The insanity rots your eyes into a pool of muck.  Good job!"
31
32 else:
33     print "You stumble around and fall on a knife and die.  Good job!"
```

WHAT YOU SHOULD SEE

C:\WINDOWS\system32\cmd.exe

```
C:\Users\UX303UB\Desktop\ewha classes 2016 SPR\Python Practices>python cthulhu_game.py
You enter a dark room with two doors.  Do you go through door #1 or door #2?
> 2
You stare into the endless abyss at Cthulhu's retina.
1. Blueberries.
2. Yellow jacket clothespins.
3. Understanding revolvers yelling melodies.
> 2
Your body survives powered by a mind of jello.  Good job!

C:\Users\UX303UB\Desktop\ewha classes 2016 SPR\Python Practices>
```

LOOPS AND LISTS: EXTREMELY IMPORTANT WHAT IS A LOOP?

```
the_count = [1, 2, 3, 4, 5]
fruits = ['apples', 'oranges', 'pears', 'apricots']
change = [1, 'pennies', 2, 'dimes', 3, 'quarters']

# this first kind of for-loop goes through a list
for number in the_count:
    print "This is count %d" % number

# same as above
for fruit in fruits:
    print "A fruit of type: %s" % fruit

# also we can go through mixed lists too
# notice we have to use %r since we don't know what's in it
for i in change:
    print "I got %r" % i

# we can also build lists, first start with an empty one
elements = []

# then use the range function to do 0 to 5 counts
for i in range(0, 6):
    print "Adding %d to the list." % i
    # append is a function that lists understand
    elements.append(i)

# now we can print them out too
for i in elements:
    print "Element was: %d" % i
```

WHILE LOOPS

```python
i = 0
numbers = []

while i < 6:
    print "At the top i is %d" % i
    numbers.append(i)

    i = i + 1
    print "Numbers now: ", numbers
    print "At the bottom i is %d" % i


print "The numbers: "

for num in numbers:
    print num
```

```
$ python ex33.py
At the top i is 0
Numbers now:  [0]
At the bottom i is 1
At the top i is 1
Numbers now:  [0, 1]
At the bottom i is 2
At the top i is 2
Numbers now:  [0, 1, 2]
At the bottom i is 3
At the top i is 3
Numbers now:  [0, 1, 2, 3]
At the bottom i is 4
At the top i is 4
Numbers now:  [0, 1, 2, 3, 4]
At the bottom i is 5
At the top i is 5
Numbers now:  [0, 1, 2, 3, 4, 5]
At the bottom i is 6
The numbers:
0
1
2
3
4
5
```

GO HOME AND TRY THIS (WITH EXIT) BRANCHES AND FUNCTIONS

```
from sys import exit

def gold_room():
    print "This room is full of gold.  How much do you take?"

    choice = raw_input("> ")
    if "0" in choice or "1" in choice:
        how_much = int(choice)
    else:
        dead("Man, learn to type a number.")

    if how_much < 50:
        print "Nice, you're not greedy, you win!"
        exit(0)
    else:
        dead("You greedy bastard!")


def bear_room():
    print "There is a bear here."
    print "The bear has a bunch of honey."
    print "The fat bear is in front of another door."
    print "How are you going to move the bear?"
    bear_moved = False

    while True:
        choice = raw_input("> ")

        if choice == "take honey":
            dead("The bear looks at you then slaps your face off.")
```

GO HOME AND TRY THIS (CONT.)

```
29          if choice == "take honey":
30              dead("The bear looks at you then slaps your face off.")
31          elif choice == "taunt bear" and not bear_moved:
32              print "The bear has moved from the door. You can go through it now."
33              bear_moved = True
34          elif choice == "taunt bear" and bear_moved:
35              dead("The bear gets pissed off and chews your leg off.")
36          elif choice == "open door" and bear_moved:
37              gold_room()
38          else:
39              print "I got no idea what that means."
40
41
42  def cthulhu_room():
43      print "Here you see the great evil Cthulhu."
44      print "He, it, whatever stares at you and you go insane."
45      print "Do you flee for your life or eat your head?"
46
47      choice = raw_input("> ")
48
49      if "flee" in choice:
50          start()
51      elif "head" in choice:
52          dead("Well that was tasty!")
53      else:
54          cthulhu_room()
55
56
57  def dead(why):
58      print why, "Good job!"
59      exit(0)
60
61  def start():
62      print "You are in a dark room."
63      print "There is a door to your right and left."
```

GO HOME AND TRY THIS (CONT.)

```
    print "You are in a dark room."
    print "There is a door to your right and left."
    print "Which one do you take?"

    choice = raw_input("> ")

    if choice == "left":
        bear_room()
    elif choice == "right":
        cthulhu_room()
    else:
        dead("You stumble around the room until you starve.")


start()
```

TRY TO UNDERSTAND THE LOGIC AND I MIGHT ASK YOU TO DISCUSS IT NEXT TIME.
IF YOU GET THIS, YOU WILL HAVE KNOWN MOST OF WHAT PEOPLE USE.

GRADIENT DESCENT/ STOCHASTIC GRADIENT DESCENT

GRADIENT DESCENT UPDATE RULE FOR WEIGHTS AND BIASES

$$w_k \rightarrow w_k' = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \rightarrow b_l' = b_l - \eta \frac{\partial C}{\partial b_l}.$$

This is the rule that's been used in gradient descent when we "role down the hill of the cost landscape.

IN PRACTICE, WE WILL FACE ONE DIFFICULTY…

$$C = \frac{1}{n} \sum_x C_x,$$

Note this averaged cost function based on the number of the training sample has this form, which is an average over C of x.

$$C_x \equiv \frac{\| y(x) - a \|^2}{2}$$

So to minimize the cost function, we need to compute the gradient for every single training input x (assuming every input is different).

BUT FOR NEURAL NETWORKS (OR DEEP LEARNING), WE SOMETIMES HAVE MILLIONS OF INPUT AT THE SAME TIME…WHICH RESULTS IN A LONG LONG COMPUTATION TIME

WHICH LEADS US TO "STOCHASTIC GRADIENT DESCENT"

Compute ONLY a small sample of X first (randomly chosen). By averaging over this small sample, we get a good measure of what the "landscape" may look like.

Like we said last time, the sample is called a "mini-batch."

STOCHASTIC GRADIENT DESCENT: MINIBATCH

Picking out a small number m of randomly chosen input data. The sample is small (10~30) but will give us the following:

$$\frac{\sum_{j=1}^{m} \nabla C_{X_j}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C, \longrightarrow \nabla C \approx \frac{1}{m} \sum_{j=1}^{m} \nabla C_{X_j},$$

Basically, you are updating weights and biases accordingly $\longrightarrow$

$$w_k \rightarrow w_k' = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$b_l \rightarrow b_l' = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l},$$

STOCHASTIC GRADIENT DESCENT: EPOCH

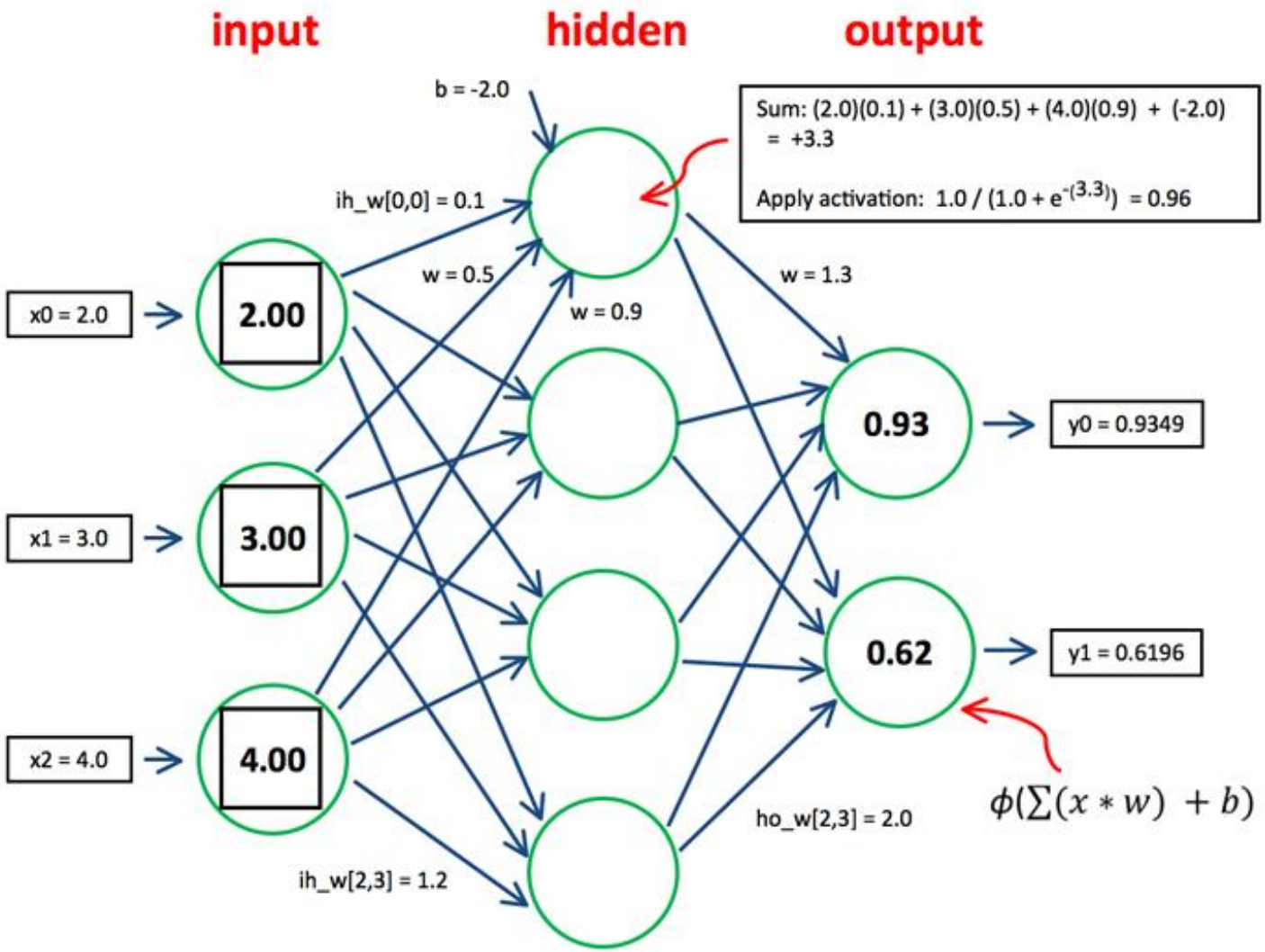
Then you pick another (not repeated) batch and keep going until you exhaust all the data => one epoch.

Essentially, you just keep updating w and b until it finishes the number of epochs of your choice.

$$w_k \rightarrow w_k' = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$b_l \rightarrow b_l' = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l},$$

ARTIFICIAL NEURAL NETWORK STRUCTURE IN A SIMPLE GRAPH



NOW LET'S LOOK AT NETWORK3

```
# Chapter 1 Codes
import mnist_loader
training_data, validation_data, test_data = mnist_loader.load_data_wrapper()

import network
net = network.Network([784, 30, 10])
net.SGD(training_data, 30, 10, 3.0, test_data=test_data)
```

```
        self.biases = [np.random.randn(y, 1) for y in sizes[1:]]
        self.weights = [np.random.randn(y, x)
                        for x, y in zip(sizes[:-1], sizes[1:])]

    def feedforward(self, a):
        """Return the output of the network if ``a`` is input."""
        for b, w in zip(self.biases, self.weights):
            a = sigmoid(np.dot(w, a)+b)
        return a

    def SGD(self, training_data, epochs, mini_batch_size, eta,
            test_data=None):
        """Train the neural network using mini-batch stochastic
```

NOW LET'S RUN THE BASE MODEL FOR MNIST DATA AGAIN

C:\WINDOWS\system32\cmd.exe - python

```
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> import mnist_loader
>>> training_data, validation_data, test_data = mnist_loader.load_data_wrapper()
>>>
>>> import network
>>> net = network.Network([784, 30, 10])
>>> net.SGD(training_data, 30, 10, 3.0, test_data=test_data)
Epoch 0: 9119 / 10000
Epoch 1: 9250 / 10000
Epoch 2: 9311 / 10000
Epoch 3: 9360 / 10000
Epoch 4: 9399 / 10000
Epoch 5: 9409 / 10000
Epoch 6: 9419 / 10000
Epoch 7: 9417 / 10000
Epoch 8: 9408 / 10000
Epoch 9: 9417 / 10000
```

HOW DO WE IMPROVE THE RESULTS?
<=> HOW DO WE REDUCE OVERFITTING

Neural Network: This is a human child ☺

HOW DO WE IMPROVE THE RESULTS? <=> HOW DO WE REDUCE OVERFITTING

Neural Network: This is a human child???

MODELS ARE USELESS IF YOU OVERFIT IT

⇒It only matters how well it "generalizes."
⇒Which means how it performs against data it's never seen before

Next week, we will discuss different techniques to avoid overfitting and increase the accuracy of the model.